## NAME

spray – scatter data in order to test the network

## SYNOPSIS

**cc** [ *flag* ... ] *file* ... **–lsocket –lnsl** [ *library* ... ]

**#include <rpcsvc/spray.h>**

**bool_t xdr_sprayarr(XDR** **xdrs*, **sprayarr** **objp***);**

**bool_t xdr_spraycumul(XDR** **xdrs*, **spraycumul** **objp***);**

## MT-LEVEL

Unsafe

## DESCRIPTION

The spray program sends packets to a given machine to test communications with that machine.

The spray program is not a C function interface, per se, but can be accessed using the generic remote procedure calling interface **clnt_call( )** (see **rpc_clnt_calls**(3N)). The program sends a packet to the called host. The host acknowledges receipt of the packet. The program counts the number of acknowledgments and can return that count.

The spray program currently supports the following procedures, which should be called in the order given:

**SPRAYPROC_CLEAR**
: This procedure clears the counter.

**SPRAYPROC_SPRAY**
: This procedure sends the packet.

**SPRAYPROC_GET**
: This procedure returns the count and the amount of time since the last **SPRAYPROC_CLEAR**.

## EXAMPLES

The following code fragment demonstrates how the spray program is used:

```
#include <rpc/rpc.h>
#include <rpcsvc/spray.h>

. . .
        spraycumul      spray_result;
        sprayarr        spray_data;
        char            buf[100];              /* arbitrary data */
        int             loop = 1000;
        CLIENT          *clnt;
        struct timeval timeout0 = {0, 0};
        struct timeval timeout25 = {25, 0};

        spray_data.sprayarr_len = (u_int)100;
        spray_data.sprayarr_val = buf;

        clnt = clnt_create("somehost", SPRAYPROG, SPRAYVERS, "netpath");
        if (clnt == (CLIENT *)NULL) {
```

```
		/* handle this error */
	}
	if (clnt_call(clnt, SPRAYPROC_CLEAR,
		xdr_void, NULL, xdr_void, NULL, timeout25)) {
			/* handle this error */
	}
	while (loop-- > 0) {
		if (clnt_call(clnt, SPRAYPROC_SPRAY,
			xdr_sprayarr, &spray_data, xdr_void, NULL, timeout0)) {
				/* handle this error */
		}
	}
	if (clnt_call(clnt, SPRAYPROC_GET,
		xdr_void, NULL, xdr_spraycumul, &spray_result, timeout25)) {
			/* handle this error */
	}
	printf("Acknowledged %ld of 1000 packets in %d secs %d usecs\n",
		spray_result.counter,
		spray_result.clock.sec,
		spray_result.clock.usec);
```

## SEE ALSO

**spray**(1M), **rpc_clnt_calls**(3N)

## NOTES

This interface is unsafe in multithreaded applications. Unsafe interfaces should be called only from the main thread.

A spray program is not useful as a networking benchmark as it uses unreliable connectionless transports, (udp for example). It can report a large number of packets dropped when the drops were caused by the program sending packets faster than they can be buffered locally (before the packets get to the network medium).